Split Recursive Least Squares: Algorithms, Architectures, and Applications

by A-Y. Wu and K.J.R. Liu

T.R. 94-37

Sponsored by
the National Science Foundation
Engineering Research Center Program,
the University of Maryland,
Harvard University,
and Industry

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **1994**	2. REPORT TYPE	3. DATES COVERED **00-00-1994 to 00-00-1994**
4. TITLE AND SUBTITLE **Split Recursive Least Squares: Algorithms, Architectures, and Applications**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Department of Electrical Engineering,Institute for Systems Research,University of Maryland,College Park,MD,20742**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT **see report**		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**33**	

Split Recursive Least Squares: Algorithms, Architectures, and Applications

An-Yeu Wu and **K. J. Ray Liu**

Electrical Engineering Department and Institute for Systems Research
University of Maryland
College Park, MD 20742
Phone: (301) 405-6619, Fax: (301) 405-6707

ABSTRACT

In this paper, a new computationally efficient algorithm for recursive least-squares (RLS) filtering is presented. The proposed *Split RLS* algorithm can perform the approximated RLS with $O(N)$ complexity for signals having no special data structure to be exploited, while avoiding the high computational complexity ($O(N^2)$) required in the conventional RLS algorithms. Our performance analysis shows that the estimation bias will be small when the input data are less correlated. We also show that for highly correlated data, the orthogonal preprocessing scheme can be used to improve the performance of the Split RLS. Furthermore, the systolic implementation of our algorithm based on the QR-decomposition RLS (QRD-RLS) array as well as its application to multidimensional adaptive filtering is also discussed. The hardware complexity for the resulting array is only $O(N)$ and the system latency can be reduced to $O(\log_2 N)$. The simulation results show that the Split RLS outperforms the conventional RLS in the application of image restoration. A major advantage of the Split RLS is its superior tracking capability over the conventional RLS under non-stationary environments.

The work is supported in part by the ONR grant N00014-93-10566 and the NSF grant MIP93-09506.

1 Introduction

The family of recursive least-squares (RLS) adaptive algorithms are well known for their superiority to the LMS-type algorithms in both convergence rate and misadjustment [1][2]. In general, the RLS algorithms do not impose any restrictions on the input data structure. As a consequence of this generality, the computational complexity is $O(N^2)$ per time iteration, where N is the size of the data matrix. This becomes the major drawback for their applications as well as for their cost-effective implementation. To alleviate the computational burden of the RLS, the family of fast RLS algorithms such as fast transversal filters, RLS lattice filters, and QR-decomposition based lattice filters (QRD-LSL), have been proposed [2]. By exploiting the special structure of the input data matrix, they can perform RLS estimation with $O(N)$ complexity. One major disadvantage of the fast RLS algorithms is that they work for data with shifting input only (*e.g.*, Toeplitz or Hankel data matrix). In many applications like multichannel adaptive array processing and image processing, the fast RLS algorithms cannot be applied because no special matrix structure can be exploited. In this paper, we propose an *approximated* RLS algorithm, which is called the *Split RLS* , based on the *projection method.* Through multiple decomposition of the signal space and making suitable approximations, we can perform RLS for non-structured data with $O(N)$ complexity. Thus, both the complexity problem in the conventional RLS and the data constraint in the fast RLS can be resolved.

The *projection method* has been used to solve large and sparse consistent linear equations such as partial differential equations (PDE). Given a linear equation $\mathbf{A}\mathbf{x} = \mathbf{b}$, where $\mathbf{A} \in \mathcal{R}^{m\times n}$, $\mathbf{b} \in \mathcal{R}^{m\times 1}$, there are two kinds of projection methods to solve it: For the consistent systems ($m = n$), the linear equation is decomposed into several smaller linear equations by "row-partitioning". Then $\mathbf{x}$ can be solved by iteration methods such as Kaczmarz projection method and Cimmino projection method [3][4][5]. For the inconsistent systems ($m > n$), $\mathbf{A}$ is decomposed into smaller submatrices by "column-partitioning". Then $\mathbf{x}$ and the residual can be solved by gradient-based iteration method [6]. Because the whole data matrix is used to compute the gradient, it is non-adaptive in nature and the convergence rate depends on the property of the $\mathbf{A}$ matrix.

In this paper, we will use the concept of column-partitioning to solve the non-structured RLS so that the computational complexity can be reduced. The signal space $\mathbf{A}$ is first partitioned into two equal-dimensional signal subspaces. After performing RLS on each subspace, we try to find an approximated optimal projection vector (of the the whole signal space) from the two optimal projection vectors of each signal subspace. Through the steps of decomposition and approximation, the complexity of the RLS can be reduced by nearly half. If now we repeatedly apply the same decomposition and approximation to each signal subspace, the RLS estimation can be solved with $O(N)$ complexity by this "divide-and-conquer" approach. We shall call such RLS estimation the

Split RLS. The systolic implementation of the Split RLS based on the QR-decomposition RLS (QRD-RLS) systolic array in [7] is also proposed. The hardware complexity for the resulting RLS array can be reduced to $O(N)$ and the system latency is only $O(\log_2 N)$.

It is noteworthy that since approximation is made while performing the Split RLS, our approach is not to obtain exact least-squares (LS) solutions. The approximation errors will introduce misadjustment (bias) to the LS errors. In order to know under what circumstances the algorithm will produce small and acceptable bias, we also provide some basic analyses for the performance of the Split RLS. The analyses together with the simulation results indicate that the Split RLS works well when applied to broad-band/less-correlated signals. Based on this observation, we also propose the *orthogonal preprocessing* scheme to improve the performance of the Split RLS. By using the transformed signals, which are less correlated than the original ones, as the inputs of the Split RLS, we can lower the bias even if the inputs are narrow-band/highly-correlated signals.

In the last part of this paper, we apply the Split RLS to the multidimensional adaptive filtering (MDAF) based on the architecture in [8]. By incorporating the well-known McClellan Transformation (MT) with the Split RLS systolic array, we can perform two-dimensional (2-D) adaptive filtering with only $O(N)$ hardware complexity and with unit throughput rate. Due to the fast convergence rate of the Split RLS, the Split RLS performs even better than the full-size QRD-RLS in the application of real-time image restoration. This also indicates that the Split RLS is preferable under non-stationary environment.

The rest of this paper is organized as follows. The projection method and the Split RLS algorithm are derived in Section 2. The systolic implementation of the proposed algorithm based on the QRD-RLS array is then described in Section 3. The performance analysis and simulation results are discussed in Section 4. An improved Split RLS algorithm using the orthogonal preprocessing scheme is considered in Section 5. Finally, the application of the Split RLS in 2-D adaptive filtering is presented in Section 6.

2 The Projection Method

Given an observation data matrix $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_n] \in \mathcal{R}^{m \times n}$ without any exhibited structure and the desired signal vector $\mathbf{y} \in \mathcal{R}^{m \times 1}$, the LS problem is to find the optimal weight coefficients

$$\hat{\mathbf{w}} = [w_1, w_2, \cdots, w_n]^T \tag{1}$$

which minimize the LS errors

$$||\mathbf{e}||^2 = ||\mathbf{A}\mathbf{w} - \mathbf{y}||^2. \tag{2}$$

In general, $\hat{\mathbf{w}}$ is of the form [9]:

$$\hat{\mathbf{w}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}. \tag{3}$$

We also have

$$\hat{\mathbf{y}} = \mathbf{A}\hat{\mathbf{w}} = \mathbf{P}\mathbf{y}, \quad \hat{\mathbf{e}} = \mathbf{y} - \hat{\mathbf{y}} \tag{4}$$

where $\hat{\mathbf{y}}$ is the optimal projection of $\mathbf{y}$ on the column space of $\mathbf{A}$, $\mathbf{P} = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ is the projection matrix, and $\hat{\mathbf{e}}$ is the optimal residual vector. The *principle of orthogonality* ensures that $\hat{\mathbf{e}}$ is orthogonal to the column space of $\mathbf{A}$. For RLS algorithms that calculate exact LS solution, such a direct projection to the N-dimensional space takes $O(N^2)$ complexity. Knowing this, in order to reduce the complexity, we shall try to perform projection onto spaces of smaller dimension.

To motivate the idea, let us consider the LS problem with the partition $\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2]$, where $\mathbf{A}_1, \mathbf{A}_2 \in \mathcal{R}^{n\times(m/2)}$. Now instead of projecting $\mathbf{y}$ directly onto the space spanned by $\mathbf{A}$ (denoted as *span*$\{\mathbf{A}\}$), we project $\mathbf{y}$ onto the two smaller subspaces, *span*$\{\mathbf{A}_1\}$ and *span*$\{\mathbf{A}_2\}$, and obtain the optimal projections $\tilde{\mathbf{y}}_1$ and $\tilde{\mathbf{y}}_2$ on each subspace (see Fig.1). The next step is to find a "good" estimation of the optimal projection $\hat{\mathbf{y}}$, say $\hat{\mathbf{y}}_{approx}$. If we can estimate a 1-D or 2-D subspace from $\tilde{\mathbf{y}}_1$ and $\tilde{\mathbf{y}}_2$ and project the desired signal $\mathbf{y}$ directly on it to obtain $\hat{\mathbf{y}}_{approx}$, the projection spaces become smaller and the computational complexity is reduced as well. There are two basic criteria for a good estimation of $\hat{\mathbf{y}}$. First, it should be in the column space of $\mathbf{A}$ matrix, *i.e.*, it must be a linear combination of the column vectors. Second, it should be as close to the real projection $\hat{\mathbf{y}}$ as possible so that the estimation error can be reduced. What worth mentioning is that since the problem itself is adaptive processing in nature, as long as $\hat{\mathbf{y}}_{approx}$will be eventually close to $\hat{\mathbf{y}}$, the initial distance between $\hat{\mathbf{y}}$ and $\hat{\mathbf{y}}_{approx}$ is not an issue. In the following, we propose two estimation methods based on their geometric relationship in the Hilbert space.

2.1 Estimation Method I (Split RLS I)

The first approach is simply to add the two subspace projections $\tilde{\mathbf{y}}_1$ and $\tilde{\mathbf{y}}_2$ together, *i.e.*,

$$\hat{\mathbf{y}}_{approx} = \tilde{\mathbf{y}}_1 + \tilde{\mathbf{y}}_2. \tag{5}$$

This provides the most intuitive and simplest way to estimate $\hat{\mathbf{y}}_{approx}$. We will show later that as $\tilde{\mathbf{y}}_1$ and $\tilde{\mathbf{y}}_2$ are more orthogonal to each other, $\hat{\mathbf{y}}_{approx}$ will approach to the optimal projection vector $\hat{\mathbf{y}}$.

Let Fig.2(a) represent one of the existing RLS algorithms that project $\mathbf{y}$ onto the N-dimensional space of $\mathbf{A}$ and compute the optimal projection $\hat{\mathbf{e}}$ (or $\hat{\mathbf{y}}$, depending on the requirements) for the current iteration. The complexity is $O(N^2)$ per time iteration for the data matrix of size N. Now using Fig.2(a) as a basic building block, we can construct the block diagram for estimation method

I as shown in Fig.2(b). Because the whole projection space is first split into two equal but smaller subspaces to perform the RLS estimation, we shall call this approach the *Split-RLS* (SP-RLS). It can be easily shown that the complexity is reduced by nearly half through such a decomposition.

The RLS algorithm based on estimation method I (SP-RLS I) can be stated as follows, where RLS($\mathbf{A}, \mathbf{y}, N$) denotes the RLS algorithm in Fig.2(a) and returns $\hat{y}(n)$ (or $\hat{e}(n)$), the last element of $\hat{\mathbf{y}}$ (or $\hat{\mathbf{e}}$), for the current iteration.

Algorithm 1 (SP-RLS I) *Given the input data vector* $\mathbf{a}(n) = [a_1(n), a_2(n), \cdots, a_N(n)]^T$ *and the desired signal* $y(n)$ *at time* n*, the SP-RLS I computes the current approximated optimal residual* $\hat{e}_{\text{approx}}(n)$ *as follows:*

SP-RLS I

1. *Update the data matrix and the desired data vector by*

$$\mathbf{A}(n) = \begin{bmatrix} \mathbf{A}(n-1) \\ \mathbf{a}^T(n) \end{bmatrix}, \mathbf{y}(n) = \begin{bmatrix} \mathbf{y}(n-1) \\ y(n) \end{bmatrix}.$$

2. *Decompose* $\mathbf{A}(n)$ *into two equal-dimensional data matrices as* $\mathbf{A}(n) = [\mathbf{A}_1(n), \mathbf{A}_2(n)]$*. Then compute the current optimal projection of each subspace by*

$$\tilde{y}_1(n) = RLS(\mathbf{A}_1(n), \mathbf{y}(n), N/2),$$
$$\tilde{y}_2(n) = RLS(\mathbf{A}_2(n), \mathbf{y}(n), N/2).$$

3. *Update the estimated optimal projection vector:*

$$\hat{\mathbf{y}}_{\text{approx}}(n) = \begin{bmatrix} \hat{\mathbf{y}}_{\text{approx}}(n-1) \\ \tilde{y}_1(n) + \tilde{y}_2(n) \end{bmatrix}.$$

4. *Project the desired signal* $\mathbf{y}(n)$ *onto the 1-D vector* $\hat{\mathbf{y}}_{\text{approx}}(n)$ *to obtain* $\hat{e}_{\text{approx}}(n)$*:*

$$\hat{e}_{\text{approx}}(n) = RLS(\hat{\mathbf{y}}_{\text{approx}}(n), \mathbf{y}(n), 1).$$

2.2 Estimation Method II (Split RLS II)

In estimation method I, we try to project $\mathbf{y}$ onto the estimated optimal projection vector $\hat{\mathbf{y}}_{approx}$. In this approach, we will project $\mathbf{y}$ directly onto the 2-D subspace $\tilde{\mathbf{A}} \triangleq span\{\tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_2\}$. As a result, the estimation shall be more accurate with slightly increase in complexity.

As with estimation method I, we can construct the block diagram for estimation method II (see

Fig.2(c)) which is similar to Fig.2(b) except for the post-processing part. The projection residual on *span*$\{\tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_2\}$ is computed through a 2-input RLS block with $\tilde{\mathbf{y}}_1$ and $\tilde{\mathbf{y}}_2$ as the inputs. The RLS algorithm based on estimation method II (SP-RLS II) is as follows:

Algorithm 2 (SP-RLS II) *Algorithm SP-RLS II is similar to the SP-RLS I except that step 3 and 4 are modified as:*

3. *Construct the n-by-2 matrix* $\tilde{\mathbf{A}}(n)$ *by*

$$\tilde{\mathbf{A}}(n) = \begin{bmatrix} \tilde{\mathbf{A}}(n-1) \\ \tilde{y}_1(n), \tilde{y}_2(n) \end{bmatrix}$$

   where $\tilde{\mathbf{A}}(0) = \mathbf{O}$.

4. *Project the desired signal* $\mathbf{y}(n)$ *onto* $\tilde{\mathbf{A}}(n)$ *to obtain* $\hat{e}_{\text{approx}}(n)$*:*

$$\hat{e}_{\text{approx}}(n) = RLS(\tilde{\mathbf{A}}(n), \mathbf{y}(n), 2).$$

2.3 Tree-Split RLS based on Estimation Method I and II

In estimation method I and II, we try to reduce the complexity by making one approximation at the last stage. Now consider the block diagram in Fig.2(c). If we repeatedly expand the two building blocks on the top by applying the same decomposition and approximation, we will obtain the block diagram in Fig.2(d). We shall call this new algorithm the *Tree-Split RLS algorithm* (TSP-RLS) due to its resemblance to a binary tree. The TSP-RLS algorithm based on Fig.2(d) is shown below.

Algorithm 3 (TSP-RLS II) *Given the input data vector* $\mathbf{a}(n) = [a_1(n), a_2(n), \cdots, a_N(n)]^T$ *and the desired signal* $y(n)$ *at time n, the TSP-RLS II computes the current approximated optimal residual* $\hat{e}_{\text{approx}}(n)$ *as follows:*

TSP-RLS II

Initialization: $\mathbf{A}_{(l)}(0) = \mathbf{O}$, *for* $l = 0, 1, \ldots, \log_2 N$, *where* $\mathbf{A}_{(l)}$ *denotes the data matrix at the* l^{th} *stage in the TSP-RLS.*

1. *Set* $l = 0$, $\mathbf{a}_{(0)}(n) = \mathbf{a}(n)$, *and update* $\mathbf{y}(n)$ *as*

$$\mathbf{y}(n) = \begin{bmatrix} \mathbf{y}(n-1) \\ y(n) \end{bmatrix}.$$

2. *Update* $\mathbf{A}_{(l)}(n)$ *as*

$$\mathbf{A}_{(l)}(n) = \begin{bmatrix} \mathbf{A}_{(l)}(n-1) \\ \mathbf{a}^T_{(l)}(n) \end{bmatrix}.$$

3. *If* $N > 2$*, compute the approximated RLS for the current stage:*

   (a) *Decompose* $\mathbf{A}_{(l)}(n)$ *into*

   $$\mathbf{A}_{(l)}(n) = [\mathbf{A}_{1,(l)}(n), \mathbf{A}_{2,(l)}(n), \ldots, \mathbf{A}_{N/2,(l)}(n)]$$

   where $\mathbf{A}_{i,(l)}(n)$ *is a n-by-2 data matrix.*

   (b) *Compute* $\tilde{y}_i(n)$ *via*

   $$\tilde{y}_i(n) = RLS(\mathbf{A}_{i,(l)}(n), \mathbf{y}(n), 2), \quad \textit{for } i = 1, 2, \ldots, N/2.$$

   (c) *Form the output vector of the current stage as* $\tilde{\mathbf{y}}_{(l)}(n) = [\tilde{y}_1(n), \tilde{y}_2(n), \ldots, \tilde{y}_{N/2}(n)]^T$.

   (d) *Set the input vector to the next stage as* $\mathbf{a}_{(l+1)}(n) = \tilde{\mathbf{y}}_{(l)}(n)$.

   (e) *Set* $l = l + 1$, $N = N/2$. *Repeat step 2-4.*

4. *Otherwise (reach the final stage), apply the RLS to compute* $\hat{e}_{\text{approx}}(n)$*:*

   $$\hat{e}_{\text{approx}}(n) = RLS(\mathbf{A}_{(l)}(n), \mathbf{y}(n), 2),$$

   and exit.

Likewise, we can derive the TSP-RLS algorithm from estimation method I (TSP-RLS I) by using the block diagram in Fig.2(b).

Lemma 1 *The computational complexity of the TSP-RLS algorithm is* $O(N)$.

Proof: For TSP-RLS II only. Let the computational complexity per time iteration for a N-input RLS be C_N. Also let $C_2 = k$ for a 2-input RLS, where k is a constant. From the block diagram in Fig.2(c), we know that the evaluation of the N-input RLS is decomposed into the evaluation of two $N/2$-input RLS's plus one 2-input RLS. Hence,

$$C_N = 2C_{N/2} + k. \tag{6}$$

Since the TSP-RLS II is obtained by recursively expanding the block diagram of the SP-RLS II, the C_N of TSP-RLS II can be computed by recursively expanding C_N in (6):

$$C_N = 2(2C_{N/4} + k) + k = \ldots = 2^l C_{N/2^l} + k \sum_{n=0}^{l-1} 2^n = (2^{l+1} - 1)k, \tag{7}$$

where l can be computed by setting $C_{N/2^l} = C_2$, *i.e.*, $l = \log_2 N - 1$. Thus,

$$C_N = (N-1)k \tag{8}$$

which is on the order of N. □

3 Systolic Implementation

In this section, we will present the systolic implementation of the above algorithms. First of all, we should note that each RLS building block in Fig.2 is independent of choices of RLS algorithms. Because the QRD-RLS array in [7] can compute the RLS estimation in a fully-pipelined way, it is a good candidate for our purpose. However, the original array computes only the optimal residual. In order to obtain the two optimal subspace projections $\tilde{\mathbf{y}}_1$ and $\tilde{\mathbf{y}}_2$, a delayed version of $y(n)$ (the desired signal at time n) should be kept in the rightmost column of the QRD-RLS array. Once the residual is computed, we can use

$$\begin{aligned} \tilde{y}_1(n) &= y(n) - \tilde{e}_1(n), \\ \tilde{y}_2(n) &= y(n) - \tilde{e}_2(n) \end{aligned} \tag{9}$$

to obtain the two subspace projections. Also, the delayed $y(n)$ can be sent to the next stage as input so that no global communication is required. Fig.3 shows the modified QRD-RLS systolic array and the detailed operations of its processing elements (PE's). In the following discussions, we shall call the modified QRD-RLS array the *projection array*, and the QRD-RLS array in [7] the *residual array*, respectively.

Now based on the block diagram in Fig.2, we can implement the Split RLS algorithms in the following way: For those RLS blocks which need to compute the optimal projection, the projection array is used for their implementations, while for those RLS blocks which need to compute the optimal residual (usually in the last stage), the residual array is used. The resulting systolic implementations of the SP-RLS I,II and the TSP-RLS II are demonstrated in Fig.4(a),(b) and (c).

Lemma 2 *The two TSP-RLS systolic arrays (TSP-RLS I,II) consist of $O(N)$ angle computers and rotators, and the total system delay is $O(\log_2 N)$.*

Proof: Suppose a N-input TSP-RLS II array requires A_N angle computers and R_N rotators. From

(6)-(8), we have

$$A_N = 2A_{N/2} + 2 = (2^{l+1} - 1)2 = 2(N-1),$$
$$R_N = 2R_{N/2} + 3 = (2^{l+1} - 1)3 = 3(N-1). \tag{10}$$

On the other hand, let the total system delay for a N-input SP-RLS II array be T_N. From Fig.4(b), we have

$$T_N = T_{N/2} + 3. \tag{11}$$

Then T_N for the TSP-RLS II can be obtained by expanding T_N in (11):

$$T_N = (T_{N/2^2} + 3) + 3 = \ldots = 3 \cdot (l+1). \tag{12}$$

Recall from Lemma 1 that $l = \log_2 N - 1$. Thus, $T_N = 3 \cdot \log_2 N$. Similarly, it can be shown that $A_N = R_N = 2N - 1$ and $T_N = 2 \cdot (\log_2 N + 1)$ for the TSP-RLS I array. □

A comparison of hardware cost for the full-size QRD-RLS in [7] (denoted as FULL-RLS), SP-RLS, TSP-RLS, and QRD-LSL [2, chap.18], is listed in Table 1. As we can see, the complexity of the TSP-RLS is comparable with the QRD-LSL which requires shift data structure.

4 Performance Analysis and Simulation Results

It is noteworthy that our approach is not an exact LS solution since the constructed $\hat{\mathbf{y}}_{approx}$ is just an approximation of the optimal projection vector. This approximation error will introduce misadjustment (bias) to the LS estimation. In the sequel, we will try to analyze the bias for SP-RLS I and SP-RLS II by investigating the relationship between the optimal projection of the whole space, $\hat{\mathbf{y}}$, and the optimal projections of the two equal-dimensional subspaces, $\tilde{\mathbf{y}}_1$ and $\tilde{\mathbf{y}}_2$. Due to the multiple RLS approximations in the TSP-RLS algorithm, it is almost impossible to provide an exact close-form solution to the final output of the TSP-RLS. Nevertheless, the analysis of the SP-RLS algorithms can give us an idea that under what conditions will the algorithms produce small and acceptable misadjustment.

4.1 Estimation Error for SP-RLS I

Consider the LS problem in (2) and decompose the column space of $\mathbf{A}$ into two equal-dimensional subspaces, *i.e.*, $\mathbf{A} = [\mathbf{A}_1\ , \mathbf{A}_2]$. Let $\hat{\mathbf{w}}^T = [\hat{\mathbf{w}}_1^T\ , \hat{\mathbf{w}}_2^T]$, then the optimal projection vector $\hat{\mathbf{y}}$ can be represented as

$$\hat{\mathbf{y}} = \mathbf{A}\hat{\mathbf{w}} = \hat{\mathbf{y}}_1 + \hat{\mathbf{y}}_2 \tag{13}$$

where $\hat{\mathbf{y}}_1 = \mathbf{A}_1\hat{\mathbf{w}}_1$ and $\hat{\mathbf{y}}_2 = \mathbf{A}_2\hat{\mathbf{w}}_2$. From the *normal equations*

$$\mathbf{A}^T\mathbf{A}\hat{\mathbf{w}} = \mathbf{A}^T\mathbf{y}, \tag{14}$$

we have

$$\mathbf{A}_1^T\mathbf{A}_1\hat{\mathbf{w}}_1 + \mathbf{A}_1^T\mathbf{A}_2\hat{\mathbf{w}}_2 = \mathbf{A}_1^T\mathbf{y}, \tag{15}$$

$$\mathbf{A}_2^T\mathbf{A}_1\hat{\mathbf{w}}_1 + \mathbf{A}_2^T\mathbf{A}_2\hat{\mathbf{w}}_2 = \mathbf{A}_2^T\mathbf{y}. \tag{16}$$

Let $\tilde{\mathbf{w}}_i, \tilde{\mathbf{y}}_i$, $i = 1, 2$, be the optimal weight vectors and the optimal projection vectors when considering two subspaces $span\{\mathbf{A}_1\}$ and $span\{\mathbf{A}_2\}$ separately. From (13) and (14), they are given by

$$\tilde{\mathbf{w}}_i = (\mathbf{A}_i^T\mathbf{A}_i)^{-1}\mathbf{A}_i^T\mathbf{y}, \quad \tilde{\mathbf{y}}_i = \mathbf{A}_i\tilde{\mathbf{w}}_i, \qquad i = 1, 2. \tag{17}$$

Premultiplying $\mathbf{A}_1(\mathbf{A}_1^T\mathbf{A}_1)^{-1}$ on (15) and using (17), we have

$$\mathbf{A}_1\hat{\mathbf{w}}_1 + \mathbf{A}_1(\mathbf{A}_1^T\mathbf{A}_1)^{-1}\mathbf{A}_1^T\mathbf{A}_2\hat{\mathbf{w}}_2 = \mathbf{A}_1\tilde{\mathbf{w}}_1. \tag{18}$$

Similarly, from (16) and (17) we can obtain

$$\mathbf{A}_2(\mathbf{A}_2^T\mathbf{A}_2)^{-1}\mathbf{A}_2^T\mathbf{A}_1\hat{\mathbf{w}}_1 + \mathbf{A}_2\hat{\mathbf{w}}_2 = \mathbf{A}_2\tilde{\mathbf{w}}_2. \tag{19}$$

By the definitions of $\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_2$, (18) and (19) can be written as

$$\hat{\mathbf{y}}_1 + \mathbf{P}_1\hat{\mathbf{y}}_2 = \tilde{\mathbf{y}}_1, \tag{20}$$

$$\mathbf{P}_2\hat{\mathbf{y}}_1 + \hat{\mathbf{y}}_2 = \tilde{\mathbf{y}}_2 \tag{21}$$

where $\mathbf{P}_i$, $i = 1, 2$ are the projection operators defined in Section 2.

In SP-RLS I, we estimate the optimal projection by

$$\hat{\mathbf{y}}_{approx} = \tilde{\mathbf{y}}_1 + \tilde{\mathbf{y}}_2, \tag{22}$$

and the estimation error (bias) is given by

$$||\Delta\mathbf{e}_1||^2 = ||\hat{\mathbf{e}}_{approx} - \hat{\mathbf{e}}||^2 = ||\hat{\mathbf{y}} - \hat{\mathbf{y}}_{approx}||^2. \tag{23}$$

Substituting (20)-(22) into (23) yields

$$||\Delta\mathbf{e}_1||^2 = ||\hat{\mathbf{y}} - \tilde{\mathbf{y}}_1 - \tilde{\mathbf{y}}_2||^2 = ||\mathbf{P}_1\hat{\mathbf{y}}_2 + \mathbf{P}_2\hat{\mathbf{y}}_1||^2. \tag{24}$$

In order to lower the bias value, $\mathbf{P}_1\tilde{\mathbf{y}}_2$ and $\mathbf{P}_2\tilde{\mathbf{y}}_1$ should be as small as possible. Note that

$$\mathbf{P}_1\tilde{\mathbf{y}}_2 = \mathbf{A}_1(\mathbf{A}_1^T\mathbf{A}_1)^{-1}\mathbf{A}_1^T\mathbf{A}_2\hat{\mathbf{w}}_2 = \mathbf{A}_1\mathbf{\Phi}_{11}^{-1}\mathbf{\Phi}_{12}\hat{\mathbf{w}}_2, \quad (25)$$

$$\mathbf{P}_2\tilde{\mathbf{y}}_1 = \mathbf{A}_2(\mathbf{A}_2^T\mathbf{A}_2)^{-1}\mathbf{A}_2^T\mathbf{A}_1\hat{\mathbf{w}}_1 = \mathbf{A}_2\mathbf{\Phi}_{22}^{-1}\mathbf{\Phi}_{21}\hat{\mathbf{w}}_1 \quad (26)$$

where $\mathbf{\Phi}_{ij} = \mathbf{A}_i^T\mathbf{A}_j$ is the deterministic correlation matrix. When the column vectors of $\mathbf{A}_1$ and $\mathbf{A}_2$ are more orthogonal to each other, $\mathbf{\Phi}_{12}$ and $\mathbf{\Phi}_{21}$ will approach to zero and the bias is reduced accordingly.

4.2 Estimation Error for SP-RLS II

Consider the block diagram of the SP-RLS II in Fig.2(c). The optimal projection of $\mathbf{y}$ onto the space *span*$\{\tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_2\}$ can be written as

$$\hat{\mathbf{y}}_{approx} = \hat{k}_1\tilde{\mathbf{y}}_1 + \hat{k}_2\tilde{\mathbf{y}}_2 \quad (27)$$

where $\hat{\mathbf{k}} = [\hat{k}_1, \hat{k}_2]^T$ is the optimal weight vector. From the *normal equations*, we have

$$[\tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_2]^T[\tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_2]\,\hat{\mathbf{k}} = [\tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_2]^T\mathbf{y}. \quad (28)$$

Using the facts that

$$\begin{array}{ll} \tilde{\mathbf{y}}_1 = \mathbf{y} - \tilde{\mathbf{e}}_1, & \tilde{\mathbf{y}}_2 = \mathbf{y} - \tilde{\mathbf{e}}_2, \\ \tilde{\mathbf{y}}_1^T\mathbf{y} = \|\tilde{\mathbf{y}}_1\|^2, & \tilde{\mathbf{y}}_2^T\mathbf{y} = \|\tilde{\mathbf{y}}_2\|^2, \end{array} \quad (29)$$

we can simplify (28) as follows:

$$\begin{cases} \hat{k}_1 + \hat{k}_2(1 - \frac{\tilde{\mathbf{y}}_1^T\tilde{\mathbf{e}}_2}{\|\tilde{\mathbf{y}}_1\|^2}) = 1 \\ \hat{k}_1(1 - \frac{\tilde{\mathbf{y}}_2^T\tilde{\mathbf{e}}_1}{\|\tilde{\mathbf{y}}_2\|^2}) + \hat{k}_2 = 1. \end{cases} \quad (30)$$

Then the optimal weight vector can be solved as

$$\hat{\mathbf{k}} = [\hat{k}_1,\ \hat{k}_2]^T = \left[\alpha\,\frac{\tilde{\mathbf{y}}_1^T\tilde{\mathbf{e}}_2}{\|\tilde{\mathbf{y}}_1\|^2},\ \alpha\,\frac{\tilde{\mathbf{y}}_2^T\tilde{\mathbf{e}}_1}{\|\tilde{\mathbf{y}}_2\|^2}\right]^T \quad (31)$$

where

$$\alpha = \left(1 - \frac{\tilde{\mathbf{y}}_1^T\tilde{\mathbf{y}}_2}{\|\tilde{\mathbf{y}}_1\|^2}\frac{\tilde{\mathbf{y}}_2^T\tilde{\mathbf{y}}_1}{\|\tilde{\mathbf{y}}_2\|^2}\right)^{-1}. \quad (32)$$

Note that $\tilde{\mathbf{y}}_1^T\tilde{\mathbf{y}}_2 = \|\tilde{\mathbf{y}}_1\|\|\tilde{\mathbf{y}}_2\|\cos\theta$, where θ denotes the angle between these two vectors, we can rewrite α as

$$\alpha = (1 - \cos^2\theta)^{-1} = \csc^2\theta. \quad (33)$$

From Fig.1, we have

$$\begin{aligned}||\hat{\mathbf{e}}_{approx}||^2 &= ||\mathbf{y}||^2 - ||\mathbf{y}_{approx}||^2 \\ &= ||\mathbf{y}||^2 - \mathbf{y}^T\mathbf{y}_{approx} \\ &= ||\mathbf{y}||^2 - \hat{k}_1||\tilde{\mathbf{y}}_1||^2 - \hat{k}_2||\tilde{\mathbf{y}}_2||^2.\end{aligned} \tag{34}$$

Substituting (31) into (34) yields

$$\begin{aligned}||\hat{\mathbf{e}}_{approx}||^2 &= ||\mathbf{y}||^2 - \csc^2\theta(\tilde{\mathbf{y}}_1^T\tilde{\mathbf{e}}_2 + \tilde{\mathbf{y}}_2^T\tilde{\mathbf{e}}_1) \\ &= ||\mathbf{y}||^2 - \csc^2\theta[\tilde{\mathbf{y}}_1^T(\mathbf{y} - \tilde{\mathbf{y}}_2) + \tilde{\mathbf{y}}_2^T(\mathbf{y} - \tilde{\mathbf{y}}_1)] \\ &= ||\mathbf{y}||^2 - \csc^2\theta||\tilde{\mathbf{y}}_1 - \tilde{\mathbf{y}}_2||^2.\end{aligned} \tag{35}$$

Thus, the bias of SP-RLS II is given by

$$\begin{aligned}||\Delta\mathbf{e}_2||^2 &= ||\hat{\mathbf{e}}_{approx}||^2 - ||\hat{\mathbf{e}}||^2 \\ &= ||\mathbf{y}||^2 - \csc^2\theta||\tilde{\mathbf{y}}_1 - \tilde{\mathbf{y}}_2||^2 - (||\mathbf{y}||^2 - ||\hat{\mathbf{y}}||^2) \\ &= ||\hat{\mathbf{y}}||^2 - \csc^2\theta||\tilde{\mathbf{y}}_1 - \tilde{\mathbf{y}}_2||^2.\end{aligned} \tag{36}$$

For any given θ, it can be shown that (see Appendix) $||\Delta\mathbf{e}_2||^2$ is bounded by

$$||\Delta\mathbf{e}_2||^2 \leq ||\Delta\mathbf{e}_1||^2. \tag{37}$$

This implies that the performance of SP-RLS II is better than that of SP-RLS I in terms of estimation error.

4.3 Bandwidth, Eigenvalue Spread, and Bias

From (24) and (36) we know that the orthogonality between the two subspaces $span\{\mathbf{A}_1\}$ and $span\{\mathbf{A}_2\}$ will significantly affect the bias value; *i.e.*, signals with different degrees of orthogonality will have different bias values for the Split RLS algorithm. However, in practice, the evaluation of degree of orthogonality for multidimensional spaces is nontrivial and computationally intensive (e.g., CS-decomposition [10, pp. 75–78]). Without loss of generality, we will only focus our discussion on single-channel case, where the data matrix **A** consists of only shifted data and the degree of orthogonality can be easily measured. In such a case, the degree of orthogonality can be measured through two indices: the bandwidth and the eigenvalue spread of the data. If the signal is less correlated (orthogonal), the autocorrelation function has smaller duration and thus larger bandwidth. Noise processes are examples. On the other hand, narrow-band processes such as sinusoidal signals are highly correlated. If the data matrix is completely orthogonal, all the eigenvalues are the same and the condition number is one. This implies that if the data matrix is more orthogonal, it will have less eigenvalue spread. It is clear from our previous discussion that the SP-RLS will render less bias for the broad-band signals than for the narrow-band signals.

As to the TSP-RLS, note that the output optimal projection is a linear combination of the input column vectors. If the inputs to one stage of the TSP-RLS array are less correlated, the outputs of this stage will still be less correlated. As an example, suppose now the inputs of the TSP-RLS II array are completely orthogonal, we have

$$\tilde{\mathbf{y}}_i = \hat{w}_{2i-1}\mathbf{a}_{2i-1} + \hat{w}_{2i}\mathbf{a}_{2i}, \quad \text{for } i = 1, 2, \ldots, N/2 \tag{38}$$

where $\hat{w}_{2i-1}$ and $\hat{w}_{2i}$ are the optimal weight coefficients in each subarray. It can be easily seen that $\tilde{\mathbf{y}}_i^T\tilde{\mathbf{y}}_j = 0$, for $i \neq j$. The orthogonality of the original inputs is still preserved at the next stage. Therefore, the signal property at the first stage such as bandwidth plays an important role in the overall performance of the TSP-RLS.

4.4 Simulation Results

In the following simulations, we will use the autoregressive (AR) process of order p (AR(p)) to generate the simulation data

$$u(n) = \sum_{i=1}^{p} w_i\, u(n-i) + v(n) \tag{39}$$

where $v(n)$ is a zero-mean white Gaussian noise with power equal to 0.1. Besides, the pole locations of the AR processes are used to control the bandwidth property: As the poles are approaching the unit circle, we will have narrow-band signals; otherwise, we will obtain broad-band signals. All the simulation results are based on the average of 100 independent trials.

In the first experiment, we try to perform fourth-order linear prediction (LP) with the AR(4) processes using the SP-RLS and TSP-RLS systolic arrays described in Section 3. In this case, the SP-RLS II is equivalent to the TSP-RLS II because they have identical implementations. Table 2 shows the AR(4) models used in this experiment. In model I and II, the two poles are at the same radii varied from 0.5 to 0.95. In model III and IV, one pole is fixed and the other is variable. For each model, the LP problem is repeated for ten times by varying the poles location from 0.5 to 0.95 with 0.05 increment. The simulation results are shown in Fig.5, in which the x-axis represents the location of the variable poles in model I-IV, and y-axis represents the average output noise power after convergence. Ideally the output should be the noise process $v(n)$ with power equal to 0.1. As we can see, when the bandwidth of input signal becomes wider, the bias is reduced. This agrees perfectly with what we expected.

Beside the bias values, we also plot the square root of the *spectral dynamic range* D (the ratio of the maximum to the minimum amplitude on the AR power spectrum) associated with each AR model. It is known that the eigenvalue spread of the data signal is bounded by the spectral dynamic range [11]

$$1 \le \frac{\lambda_{\max}}{\lambda_{\min}} \le \frac{\max\{|U(e^{j\omega})|^2\}}{\min\{|U(e^{j\omega})|^2\}} \triangleq D, \tag{40}$$

where $U(e^{j\omega})$ is the spectrum of $u(n)$. From the simulation results, we see the consistency between the bias value and the spectral dynamic range. This indicates that the performance of the Split RLS algorithms is also affected by the eigenvalue spread of the input signal. This phenomenon is similar to what we have seen in the LMS-type algorithms.

In the second experiment, we extend the previous experiment to perform eighth-order LP for four AR(8) processes. The setting for the poles is listed in Table 3. The simulation results, shown in Fig.6, again validate the bandwidth-bias relationship. Beside the bias effect, two observations can be made from these two experimental results:

1. The SP-RLS performs better than the TSP-RLS. This is pretty much due to the number of approximation stages in each algorithm.

2. The overall performance of SP-RLS II is better than that of SP-RLS I. This agrees with our analysis in (37).

Next we want to examine the convergence rate of our algorithm. An AR(7) model is used to generate data and the sum of the current inputs is used as the desired signal. The output should be zero after it converges. Fig.7 shows the convergence curve for the 8-input FULL-RLS and the TSP-RLS II after some initial perturbation. It is interesting to note that although the TSP-RLS II has some bias after it converges, its convergence rate is faster than that of the FULL-RLS. This is due to the fact that the $O(\log_2 N)$ system latency of the TSP-RLS is less than the $O(N)$ latency of the FULL-RLS. Also, to initialize an 8-input full-size array takes more time than to initialize the three small cascaded 2-input arrays. The property of faster convergence rate is especially preferred for the tracking of parameters in non-stationary environments. In Section 6 we will provide an image restoration simulation to verify this observation.

5 Projection Method with Orthogonal Preprocessing

In the previous sections, we have seen that the Split RLS performs very well when the input signal is less correlated (or broad-band). However, in many applications, processing of highly-correlated (or narrow-band) signals is inevitable. We are thus motivated to investigate a way to improve the Split RLS algorithm when dealing with highly-correlated signals. From the analyses in Section 4, we know that the estimated optimal projection will approach to the real optimal projection when all subspaces are more orthogonal to each other. Therefore, if we can preprocess the data matrix such that the column spaces become more orthogonal (less correlated) to each other. a better performance is expected. Such a concept has been employed in the "Transform domain LMS algorithm" (TDLMS) [12][13][14], as well as in the row-partitioning projection methods [3][4]. It is clear that Gram-Schmidt orthogonalization will render an excellent performance. However, the $O(N^2)$ complexity prevents us from considering it.

5.1 Transform-Domain LS Problem

In transform-domain signal processing, the input data matrix $\mathbf{A}$ is first transformed into another data matrix $\mathbf{Z}$

$$\mathbf{Z} = \mathbf{A}T \tag{41}$$

where T is an unitary transformation matrix of rank N. The transform-domain LS problem is to find the optimal weight vector $\hat{\mathbf{k}} = [k_1, k_2, \cdots, k_N]^T$ which minimizes the LS error $||\mathbf{Zk} - \mathbf{y}||^2$ in the transform domain. Because $\mathbf{Z}$ and $\mathbf{A}$ span the same signal space, the LS error will be the same as in (2). The transformation process can be viewed as a set of filter banks with equally spaced mainlobes [14]. Each column vector of $\mathbf{Z}$ corresponds to the output signal of a given filter in the filter banks. Therefore, the column vectors of $\mathbf{Z}$ will be less correlated than those of $\mathbf{A}$. This helps us to obtain a better $\hat{\mathbf{y}}_{approx}$ according to our observations in (24) and (36).

The operation for the Split RLS with orthogonal preprocessing is as follows: First perform the orthogonal transform on the current data vector, then use the transformed data as the inputs of the Split RLS. In our approach, the Discrete Cosine Transform (DCT) and the Discrete Hartley Transform (DHT) are used as the preprocessing kernels. As to the hardware implementation, we can employ the time-recursive DCT/DHT lattice structure in [15] to continuously generate the transformed data. Fig.8 shows the SP-RLS I array with DCT/DHT preprocessing. The transform-domain data are first generated through the DCT/DHT lattice structure, then are sent to the SP-RLS I array to perform the RLS filtering. The TSP-RLS array with the preprocessing scheme can be constructed in a similar way. Since both the DCT/DHT lattice structure and the TSP-RLS array require $O(N)$ hardware complexity, the total cost for the whole system is still $O(N)$.

In addition to the two aforementioned transforms, for the purpose of further decorrelation, we also propose a new preprocessing scheme called the *Swapped DCT* (SWAP-DCT) based on the DCT. Suppose $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N]$ is the DCT-domain data. In the DCT preprocessing given in Fig.8, the input data is partitioned as

$$\begin{aligned} \mathbf{A}_1 &= [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{N/2}], \\ \mathbf{A}_2 &= [\mathbf{z}_{N/2+1}, \mathbf{z}_{N/2+2}, \cdots, \mathbf{z}_N]. \end{aligned} \tag{42}$$

To make the input data more uncorrelated, we permute the transformed data column as

$$\begin{aligned} \mathbf{A}_1 &= [\mathbf{z}_1, \mathbf{z}_3, \ldots, \mathbf{z}_{2k-1}, \ldots, \mathbf{z}_{N-1}], \\ \mathbf{A}_2 &= [\mathbf{z}_2, \mathbf{z}_4, \ldots, \mathbf{z}_{2k}, \ldots, \mathbf{z}_N] \end{aligned} \tag{43}$$

in the SWAP-DCT preprocessing scheme. Fig.9 shows the spectrum of the normal DCT partitioning and the SWAP-DCT partitioning. Recall that the eigenvalue spread will affect the bias value, and the eigenvalue spread is bounded by the spectral dynamic range. It is obvious that the SWAP-DCT preprocessing scheme will have better performance due to the smaller eigenvalue spread in both $\mathbf{A}_1$ and $\mathbf{A}_2$.

5.2 Simulation of the TSP-RLS with Orthogonal Preprocessing

To validate our arguments for the orthogonal preprocessing, we will repeat the two experiments in Section 4.4 for the TSP-RLS II with three different preprocessing schemes (DCT, DHT, SWAP-DCT). The simulation results are given in Fig.10 and Fig.11. In general, the TSP-RLS with DCT preprocessing gives a fairly significant improvement in the bias value over the TSP-RLS without any preprocessing (Normal TSP-RLS). Nevertheless, some exceptions can be found in AR(4).III and AR(8).III. As to the DHT, it does not perform well in most cases except in AR(8).II and AR(8).IV. It is as expected that the SWAP-DCT performs better than the DCT in most cases. This supports our assertion for the effect of the SWAP-DCT.

From the simulation results, we can see that it is almost impossible to find one transform that is optimal for all signals. This is also true for the TDLMS algorithms [13]. In general, the DCT and the SWAP-DCT are good choices to improve the performance.

6 Application to Multidimensional Adaptive Filtering

In this section, we will apply the Split RLS to the multidimensional adaptive filtering (MDAF) based on the architecture in [8]. In [8], the McClellan Transformation (MT) [16] was employed to reduce the total parameters in the 2-D filter design, and the QRD-RLS array in [17] was used as the processing kernel to update the weight coefficients. In our approach, we replace the QRD-RLS array with the Split RLS array. This will result in a more cost-effective MDAF architecture while with even better performance.

6.1 2-D Adaptive Filtering using McClellan Transformation

Given a 1-D zero-phase FIR filter with support $-N \leq i \leq N$, the frequency response can be written as

$$H(\omega) = \sum_{i=0}^{N} h_i \cos(i\omega) = \sum_{i=0}^{N} h_i T_i[\cos\omega]. \tag{44}$$

where $T_i[\cdot]$ denotes the Chebyshev polynomial of degree i. Using the transformation of variables [16]

$$F(\omega_1, \omega_2) \longrightarrow \cos\omega, \tag{45}$$

we obtain the MT 2-D frequency response

$$H(\omega_1, \omega_2) = \sum_{i=0}^{N} h_i T_i[F(\omega_1, \omega_2)]. \tag{46}$$

The MT is a near-optimal design method for 2-D filters [18, chap.4]. It decomposes the design problem into the design of the *1-D prototype FIR filter*, h_i, $i = 0, 1, \cdots, N$, and the *2-D transformation function*, $F(\omega_1, \omega_2)$. The former defines the frequency response along the 2-D frequency plane, while the latter, which is usually a small fixed 2-D zero-phase FIR filter, maps the 1-D frequencies into contours in the 2-D frequency plane. Fig.12 shows the block diagram which performs 2-D filtering based on the MT and the Chebyshev recursion [19][8]. Each PE is a linear systolic array realizing the 2-D transformation function in (45) with $x_i(n_1, n_2), i = 0, 1, \cdots, N$, as the PE output. $y(n_1, n_2)$ is the desired 2-D signal, and $\mathbf{h} = [h_1, h_2, \cdots, h_N]^T$ is the tap coefficient vector of the 1-D prototype filter. In [8], **h** is updated by considering Fig.12 as a multichannel LS problem, *i.e.*, **h** is obtained by minimizing the LS error

$$||e(n_1, n_2)||^2 = ||y(n_1, n_2) - \hat{y}(n_1, n_2)||^2 = ||y(n_1, n_2) - \sum_{i=0}^{N} h_i x_i(n_1, n_2)||^2 \tag{47}$$

for each incoming data. For the systolic implementation, **h** is solved through the QRD-RLS array in [17] with $x_i(n_1, n_2)$'s and $y(n_1, n_2)$ as the array inputs. However, the opposite data wavefront in the QRD-RLS array as well as the $O(N^2)$ hardware complexity makes the system inappropriate for cost-effective pipelined processing.

In some applications, such as image restoration and image registration, the estimation error $e(n_1, n_2)$ is the only parameter of interest. In such a case, we can modify the MDAF structure in [8] by replacing the QRD-RLS array with the FULL-RLS array since the latter produces the LS error in a fully-pipelined way. To further reduce the hardware complexity, we can employ the TSP-RLS array as the processing kernel. As a result, we can perform 2-D adaptive filtering with $O(N)$ hardware complexity and with unit throughput rate.

6.2 Simulation with TDALE

The performance of the proposed MDAF architecture is examined by applying it to a two-dimensional adaptive line enhancer (TDALE) [20][21] for image restoration. The block diagram is depicted in Fig.13. The primary input is the well-known "LENA" image degraded by a white Gaussian noise. A 2-D unit delay $z_1^{-1}z_2^{-1}$ is used as a decorrelation operator to obtain the reference image. The image signal is fed into the system in the raster scanned format - from left to right, top to bottom. After the input image goes through the TSP-RLS array, the generated estimation error is subtracted from the reference signal to get the filtered image. For comparison, we also repeat this experiment using the FULL-RLS array.

The simulation results are shown in Table 4 and in Fig.14. We can see that the performance of the TSP-RLS is better than the 2-D joint process lattice structure in [21] when the signal-to-noise ratio (SNR) is low. It is also interesting to note that the TSP-RLS outperforms the FULL-RLS. As we discussed in Section 4.4, although the TSP-RLS has misadjustment after convergence, it converges faster than the FULL-RLS. This fast-tracking property is preferable under non-stationary environments where convergence is very unlikely.

7 Conclusions

In this paper, we introduced a new $O(N)$ fast algorithm and architecture for the RLS estimation of nonstructured data. Compared with the conventional RLS, this new approach is sub-optimal in the sense that it introduces extra bias to the LS estimations. Nevertheless, we have shown that the bandwidth and/or the eigenvalue spread of the input signal can be used as a good performance index for these algorithms. Therefore, the users will have small bias when dealing with broad-band/less-correlated signals. For narrow-band signals, we can also employ the orthogonal preprocessing to improve its performance. The low complexity as well as the fast convergence rate of the proposed algorithm makes it suitable for RLS estimation under the non-stationary or fast-changing environments where the data matrix has no structure. For example, one possible application of the Split RLS is in the Sidelobe Cancellor (SLC), in which the inputs of the auxiliary arrays are mainly noises. The fast tracking capability of the Split RLS algorithm, as demonstrated in the image restoration simulations, provides a very promising potential for parameter tracking under non-stationary environments. Furthermore, the systolic architecture of the Split RLS is fully parallel and pipelined and thus provides a high-throughput implementation for real-time applications.

Appendix

In this appendix, we will show that the bias of SP-RLS II is bounded by that of SP-RLS I. From (24) and (36), we have

$$\begin{aligned} \|\Delta \mathbf{e}_1\|^2 &= \|\hat{\mathbf{y}} - \tilde{\mathbf{y}}_1 - \tilde{\mathbf{y}}_2\|^2, \\ \|\Delta \mathbf{e}_2\|^2 &= \|\hat{\mathbf{y}}\|^2 - \csc^2\theta \|\tilde{\mathbf{y}}_1 - \tilde{\mathbf{y}}_2\|^2. \end{aligned} \tag{48}$$

Note that $\csc^2\theta \geq 1$ for any θ. Thus,

$$\begin{aligned} \|\Delta \mathbf{e}_1\|^2 - \|\Delta \mathbf{e}_2\|^2 &\geq \|\hat{\mathbf{y}} - \tilde{\mathbf{y}}_1 - \tilde{\mathbf{y}}_2\|^2 - \|\hat{\mathbf{y}}\|^2 + \|\tilde{\mathbf{y}}_1 - \tilde{\mathbf{y}}_2\|^2 \\ &= 2[(\|\tilde{\mathbf{y}}_1\|^2 + \|\tilde{\mathbf{y}}_2\|^2) - \hat{\mathbf{y}}^T(\tilde{\mathbf{y}}_1 + \tilde{\mathbf{y}}_2)]. \end{aligned} \tag{49}$$

From (20) and (21), we have

$$\|\tilde{\mathbf{y}}_1\|^2 = \|\hat{\mathbf{y}}_1\|^2 + 2\hat{\mathbf{y}}_1^T\hat{\mathbf{y}}_2 + \hat{\mathbf{y}}_2^T(\mathbf{P}_1\hat{\mathbf{y}}_2), \tag{50}$$

$$\|\tilde{\mathbf{y}}_2\|^2 = \|\hat{\mathbf{y}}_2\|^2 + 2\hat{\mathbf{y}}_1^T\hat{\mathbf{y}}_2 + \hat{\mathbf{y}}_1^T(\mathbf{P}_2\hat{\mathbf{y}}_1) \tag{51}$$

where the fact that $\hat{\mathbf{y}}_1^T(\mathbf{P}_1\hat{\mathbf{y}}_2) = \hat{\mathbf{y}}_2^T(\mathbf{P}_2\hat{\mathbf{y}}_1) = \hat{\mathbf{y}}_1^T\hat{\mathbf{y}}_2$ is used. Combining (50) and (51) yields

$$\|\tilde{\mathbf{y}}_1\|^2 + \|\tilde{\mathbf{y}}_2\|^2 = \|\hat{\mathbf{y}}\|^2 + 2\,\hat{\mathbf{y}}_1^T\hat{\mathbf{y}}_2 + \hat{\mathbf{y}}_2^T(\mathbf{P}_1\hat{\mathbf{y}}_2) + \hat{\mathbf{y}}_1^T(\mathbf{P}_2\hat{\mathbf{y}}_1). \tag{52}$$

On the other hand,

$$\hat{\mathbf{y}}^T(\tilde{\mathbf{y}}_1 + \tilde{\mathbf{y}}_2) = \hat{\mathbf{y}}^T[(\hat{\mathbf{y}}_1 + \mathbf{P}_1\hat{\mathbf{y}}_2) + (\hat{\mathbf{y}}_2 + \mathbf{P}_2\hat{\mathbf{y}}_1)] = \|\hat{\mathbf{y}}\|^2 + \hat{\mathbf{y}}^T(\mathbf{P}_1\hat{\mathbf{y}}_2) + \hat{\mathbf{y}}^T(\mathbf{P}_2\hat{\mathbf{y}}_1). \tag{53}$$

Substituting (52) and (53) into (49), we have

$$\|\Delta \mathbf{e}_1\|^2 - \|\Delta \mathbf{e}_2\|^2 \geq 2\,[2\,\hat{\mathbf{y}}_1^T\hat{\mathbf{y}}_2 - \hat{\mathbf{y}}_1^T(\mathbf{P}_1\hat{\mathbf{y}}_2) - \hat{\mathbf{y}}_2^T(\mathbf{P}_2\hat{\mathbf{y}}_1)] = 2\,[(\hat{\mathbf{y}}_1^T(\hat{\mathbf{y}}_2 - \mathbf{P}_1\hat{\mathbf{y}}_2) + \hat{\mathbf{y}}_2^T(\hat{\mathbf{y}}_1 - \mathbf{P}_2\hat{\mathbf{y}}_1)]. \tag{54}$$

Because $\mathbf{P}_1$ and $\mathbf{P}_2$ are projection matrices, it is clear that $\hat{\mathbf{y}}_2 \geq \mathbf{P}_1\hat{\mathbf{y}}_2$, $\hat{\mathbf{y}}_1 \geq \mathbf{P}_2\hat{\mathbf{y}}_1$. Therefore, $\|\Delta \mathbf{e}_1\|^2 - \|\Delta \mathbf{e}_2\|^2 \geq 0$, *i.e.*, $\|\Delta \mathbf{e}_2\|^2 \leq \|\Delta \mathbf{e}_1\|^2$. $\square$

References

[1] M. L. Honig and D. G. Messerschmitt, *Adaptive Filters : Structures, Algorithms, and Applications.* Kluwer Academic Publishers, 1984.

[2] S. Haykin, *Adaptive Filter Theory.* Prentice-Hall, Englewood Cliffs, N.J., 2nd ed., 1991.

[3] A. S. Kydes and R. P. Tewarson, "An iterative methods for solving partitioned linear equations," *Computing*, vol. 15, pp. 357–363, Jan. 1975.

[4] T. Elfving, "Block-iterative methods for consistent and inconsistent linear equations," *Numer. Math.*, vol. 35, pp. 1–12, 1980.

[5] R. Bramley and A. Samem, "Row projection methods for large nonsymmetric linear systems," *SIAM J. Sci. Stat. Comput.*, vol. 13, pp. 168–193, Jan. 1992.

[6] K. Tanabe, "Projection method for solving a singular system of linear equations and its applications," *Numer. Math.*, vol. 17, pp. 203–214, 1971.

[7] J. G. McWhirter, "Recursive least-squares minimization using a systolic array.," *Proc. SPIE, Real-Time Signal Processing VI*, vol. 431, pp. 105–112, 1983.

[8] J. M. Shapiro and D. H. Staelin, "Algorithms and systolic architecture for multidimensional adaptive filtering via McClellan transformation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, pp. 60–71, Mar 1992.

[9] G. W. Stewart, *Introduction to Matrix Computations.* Academic Press, New York, 1973.

[10] G. H. Golub and C. F. Van Loan, *Matrix Computations.* The John Hopkins University Press, Baltimore, MD, 2nd ed., 1989.

[11] J. Makhoul, "Linear Prediction: A tutorial review," *Proc. IEEE*, vol. 63, pp. 561–580, April 1975.

[12] S. S. Narayan, A. M. Peterson, and M. J. Narasimha, "Transform domain LMS algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 31, pp. 609–615, June 1983.

[13] D. F. Marshall, W. K. Jenkins, and J. J. Murphy, "The use of orthogonal transforms for improving performance of adaptive filters," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 474–484, April 1989.

[14] B. Farhang-Boroujeny and S. Gazor, "Selection of orthonormal transforms for improving the performance of the transform domain normalised LMS algorithm," *IEE Proceedings-F*, vol. 139, pp. 327–335, Oct. 1992.

[15] K. J. R. Liu and C. T. Chiu, "Unified parallel lattice structures for time-recursive Discrete Cosine/Sine/Hartley transforms," *IEEE Trans. Signal Processing*, vol. 41, pp. 1357–1377, March 1993.

[16] R. M. Mersereau, W. F. G. Mecklenbrauker, and T. F. Quatieri, Jr., "McClellan transformations for two-dimensional digital filtering: I - Design," *IEEE Trans. Circuits Syst.*, vol. 23, pp. 405–422, July 1976.

[17] W. M. Gentleman and H. T. Kung, "Matrix triangularization by systolic arrays," *Proc. SPIE, Real-Time Signal Processing IV*, vol. 298, pp. 298–303, 1981.

[18] J. S. Lim, *Two-dimensional signal and image processing.* Englewood Cliffs, New Jersey: Prentice-Hall, 1990.

[19] J. H. McClellan and D. S. K. Chan, "A 2-D FIR filter structure derived form the Chebyshev recursion," *IEEE Trans. Circuits Syst.*, vol. 24, pp. 372–378, July 1977.

[20] M. M. Hadhoud and D. W. Thomas, "The two-dimensional adaptive LMS (TDLMS) algorithm," *IEEE Trans. Circuits Syst.*, vol. 5, pp. 485–494, May 1988.

[21] H. Youlal, Malika Janati-I, and M. Najim, "Two-dimensional joint process lattice for adaptive restoration of images," *IEEE Trans. Image Processing*, vol. 1, pp. 366–378, July 1992.

	No. of Angle Computers	No. of Rotators	System latency
FULL-RLS	N	$N(N+1)/2$	$N+1$
SP-RLS I	$N+1$	$N^2/4+N/2+1$	$N/2+3$
SP-RLS II	$N+2$	$N^2/4+N/2+3$	$N/2+4$
TSP-RLS I	$2N-1$	$2N-1$	$2(\log_2 N+1)$
TSP-RLS II	$2(N-1)$	$3(N-1)$	$3\log_2 N$
QRD-LSL	$2N+1$	$3N+1$	$N+1$

Table 1: Comparison of hardware cost for the FULL-RLS, SP-RLS, TSP-RLS, and QRD-LSL, where the QRD-LSL requires shift data structure.

AR(4)	ρ_1	ρ_2	ϕ_1	ϕ_2
I	0.5 - 0.95	ρ_1	$1/8\pi$	$4/8\pi$
II	0.5 - 0.95	ρ_1	$5/8\pi$	$7/8\pi$
III	0.5 - 0.95	0.6	$1/8\pi$	$4/8\pi$
IV	0.6	0.5 - 0.95	$5/8\pi$	$7/8\pi$

Table 2: List of the AR(4) models used in Experiment 1 (with poles at $\rho_1 e^{\pm j\phi 1}$ and $\rho_2 e^{\pm j\phi 2}$).

AR(8)	ρ_1	ρ_2	ρ_3	ρ_4	ϕ_1	ϕ_2	ϕ_3	ϕ_4
I	0.5 - 0.95	ρ_1	ρ_1	ρ_1	$1/15\pi$	$4/15\pi$	$8/15\pi$	$12/15\pi$
II	0.5 - 0.95	ρ_1	ρ_1	ρ_1	$2/15\pi$	$5/15\pi$	$8/15\pi$	$11/15\pi$
III	0.5 - 0.95	0.6	ρ_1	0.6	$1/15\pi$	$4/15\pi$	$8/15\pi$	$12/15\pi$
IV	0.6	0.5 - 0.95	0.6	ρ_2	$2/15\pi$	$5/15\pi$	$8/15\pi$	$11/15\pi$

Table 3: List of the AR(8) models used in Experiment 2 (with poles at $\rho_1 e^{\pm j\phi 1}$, $\rho_2 e^{\pm j\phi 2}$, $\rho_3 e^{\pm j\phi 3}$, $\rho_4 e^{\pm j\phi 4}$).

Input SNR (dB)	10.0	3.0	0.0
Output SNR in [21]	12.0	8.0	6.0
Output SNR using FULL-RLS	10.5	9.0	7.6
Output SNR using TSP-RLS II	10.9	9.8	8.7

Table 4: SNR results of the TDALE in the application of restoring noisy image.

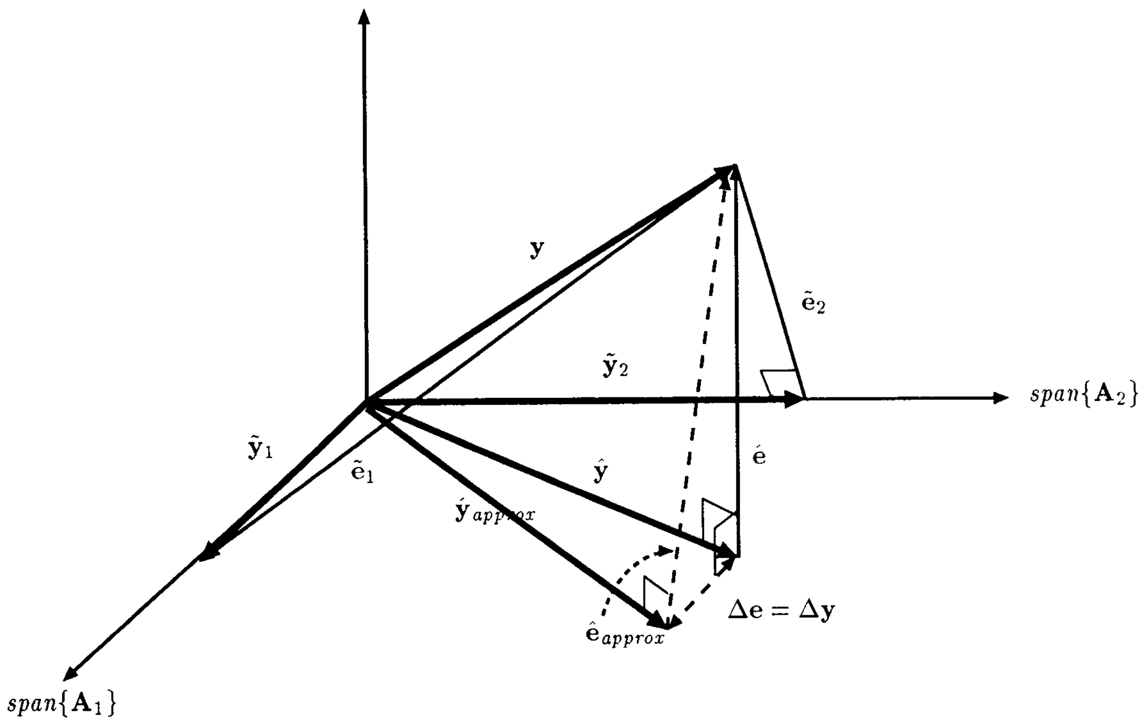
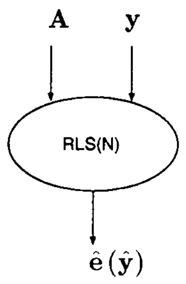
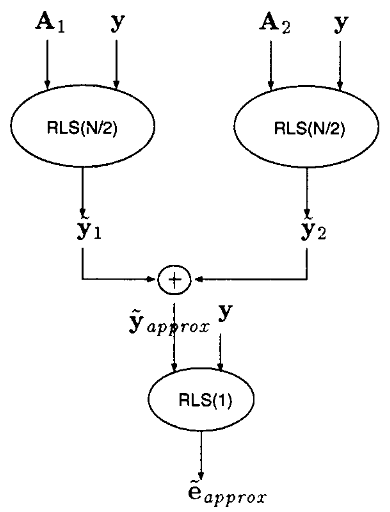


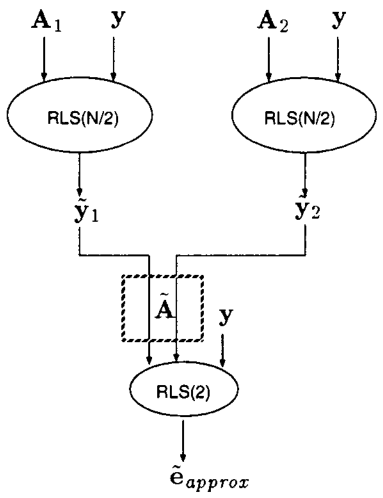
Figure 1: Geometric interpretation of the projection method.



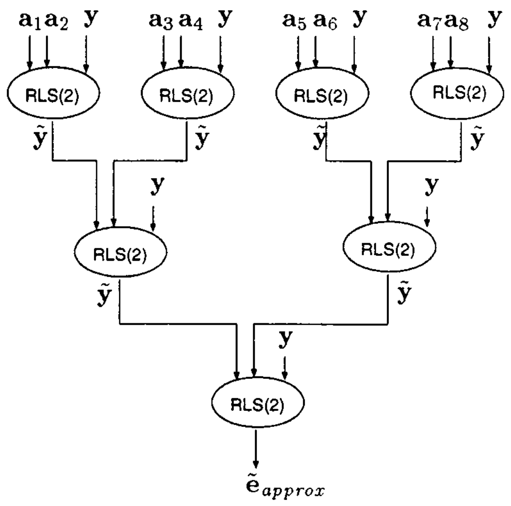
(a)



(b)

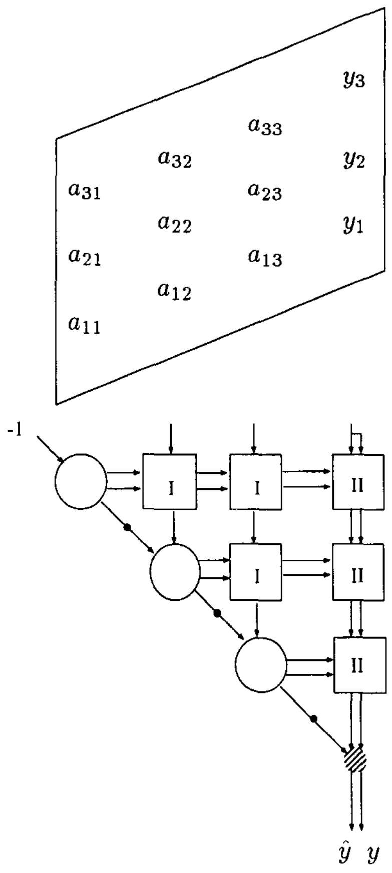


(c)



(d)

Figure 2: Block diagram for (a) a N-input RLS algorithm, (b) the SP-RLS I algorithm, (c) the SP-RLS II algorithm, (d) the TSP-RLS II algorithm.



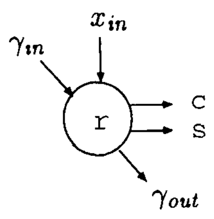
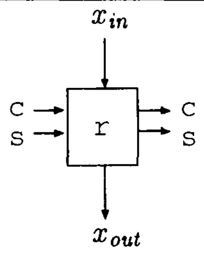
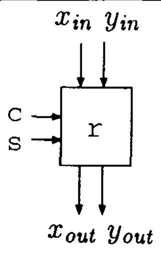
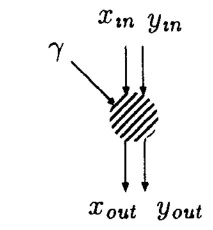
	Angle Computers	Rotators I	Rotators II	Modified Multiplier
PE				
PE Operation	If $x_{in} = 0$ then $c \leftarrow 1$; $s \leftarrow 0$; otherwise $r' = \sqrt{r^2 + x_{in}^2}$ $c \leftarrow r/r'$; $s \leftarrow x_{in}/r'$ $r \leftarrow r'$ end $\gamma_{out} \leftarrow c\gamma_{in}$	$x_{out} \leftarrow cx_{in} - sr$ $r \leftarrow sx_{in} + cr$	$x_{out} \leftarrow cx_{in} - sr$ $r \leftarrow sx_{in} + cr$ $y_{out} \leftarrow y_{in}$	$x_{out} \leftarrow y_{in} - \gamma x_{in}$ $y_{out} \leftarrow y_{in}$

Figure 3: Modified QRD-RLS array (Projection array) and its PE operations.

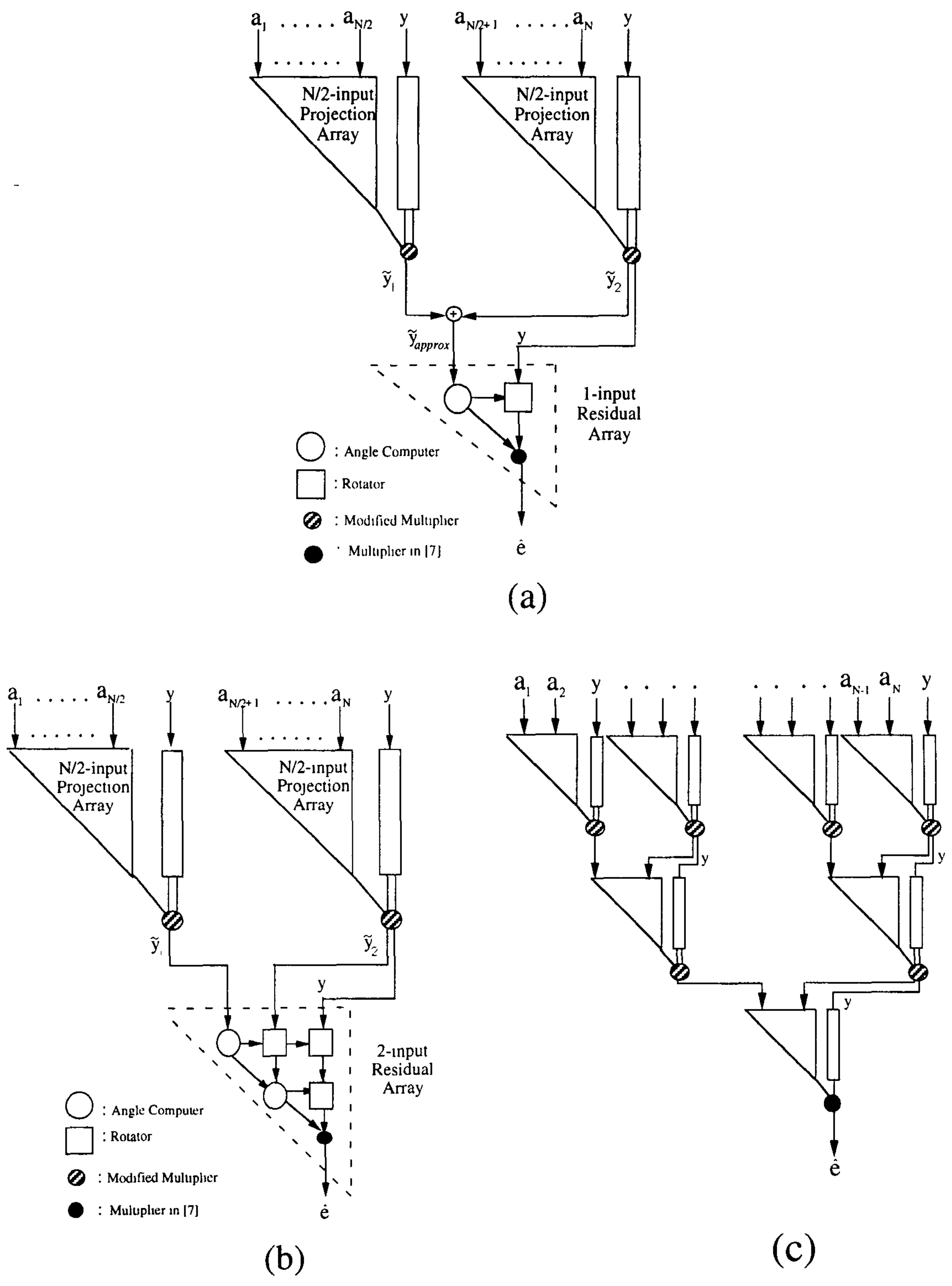


Figure 4: Systolic implementation of (a) the SP-RLS I, (b) the SP-RLS II, (c) the TSP-RLS II.

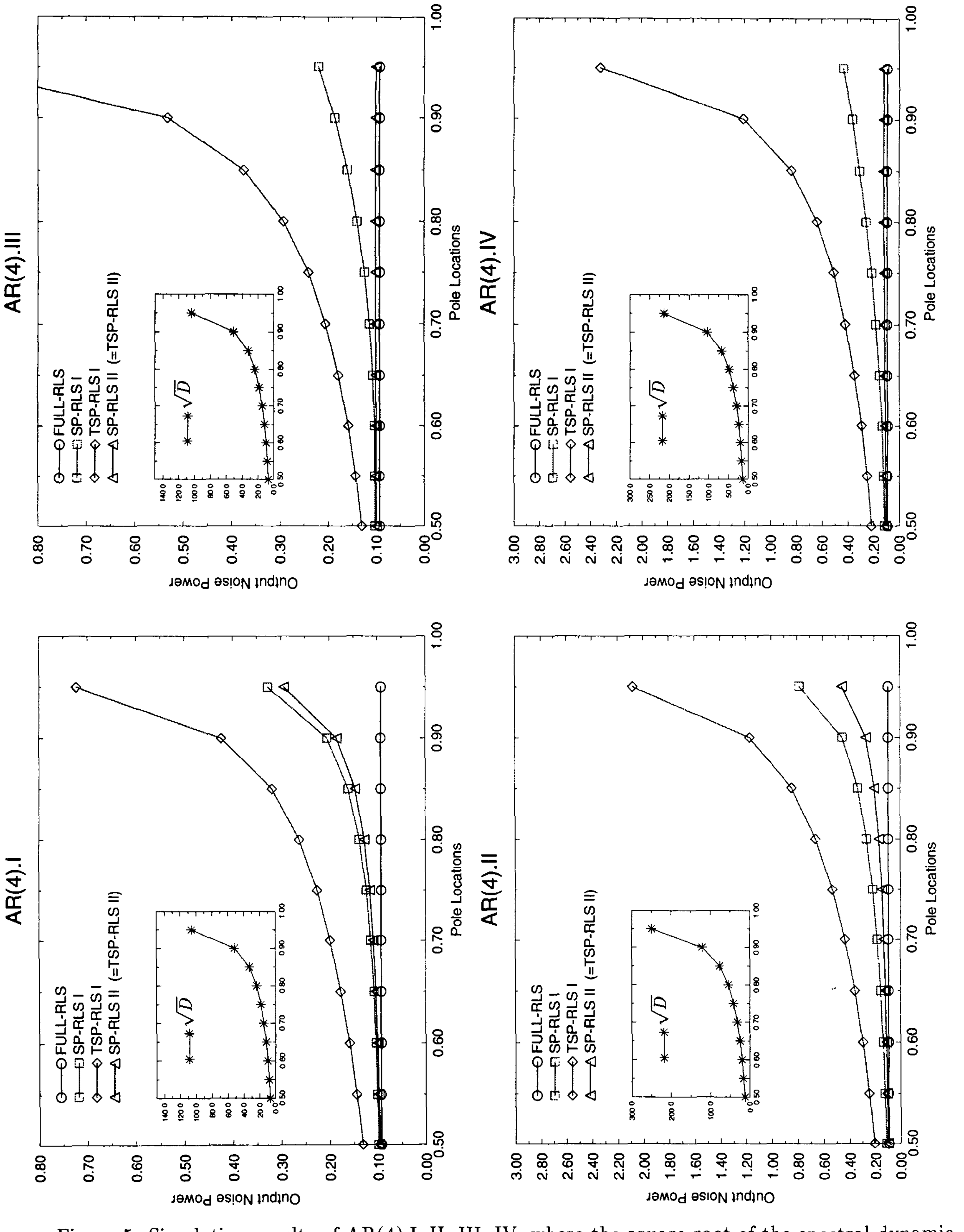


Figure 5: Simulation results of AR(4).I, II, III, IV, where the square root of the spectral dynamic range (D) is also plotted for comparison.

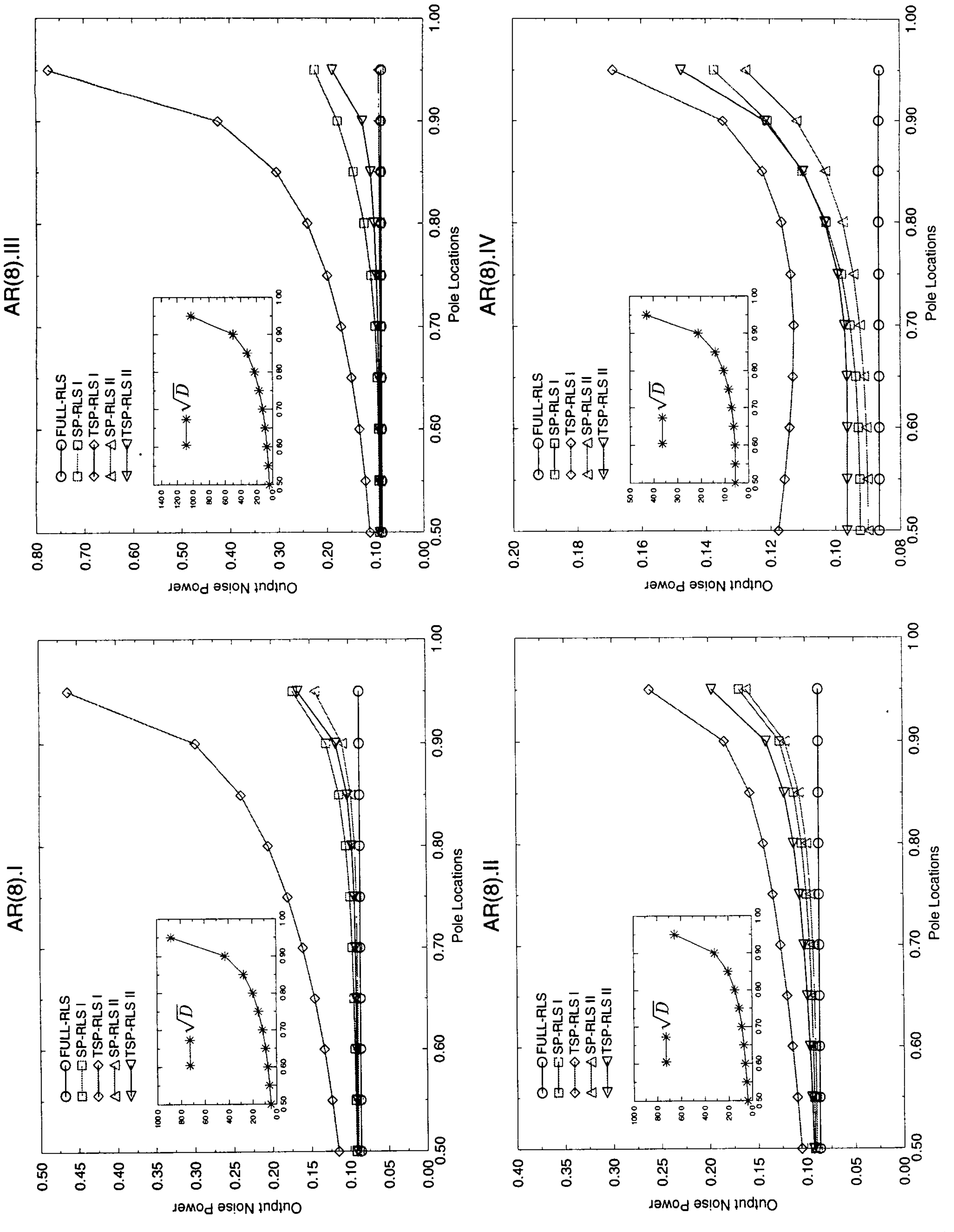


Figure 6: Simulation results of AR(8).I, II, III, IV, where the square root of the spectral dynamic range (D) is also plotted for comparison.

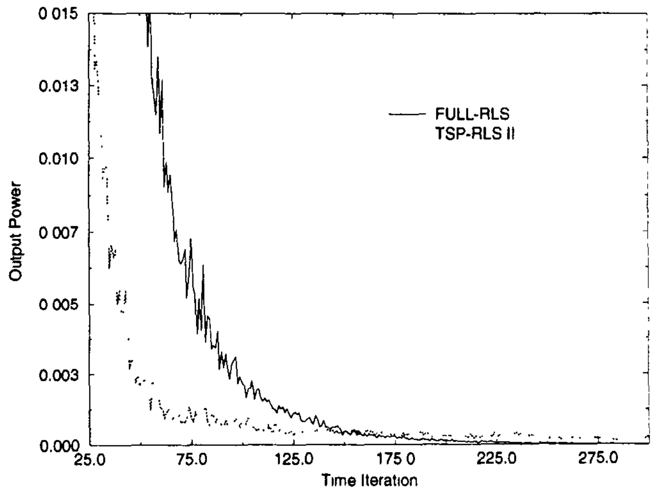


Figure 7: Learning curve of the FULL-RLS and TSP-RLS II after some initial perturbation.

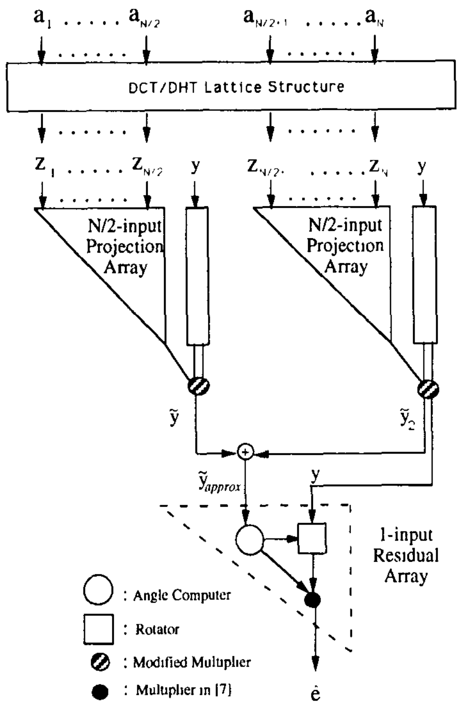


Figure 8: SP-RLS I array with orthogonal preprocessing.

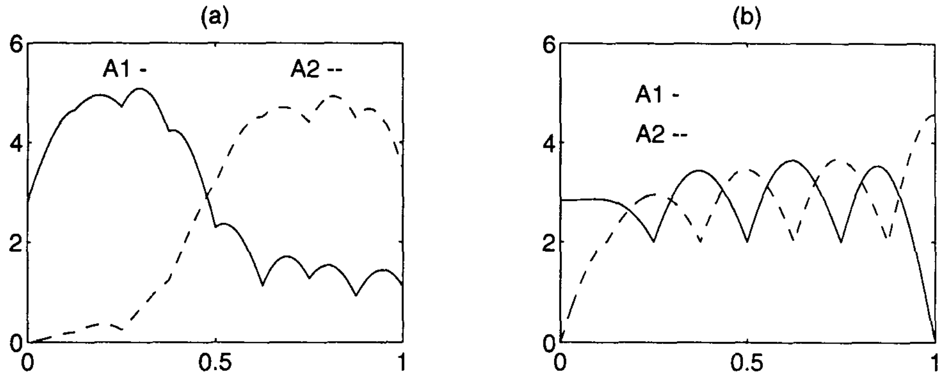
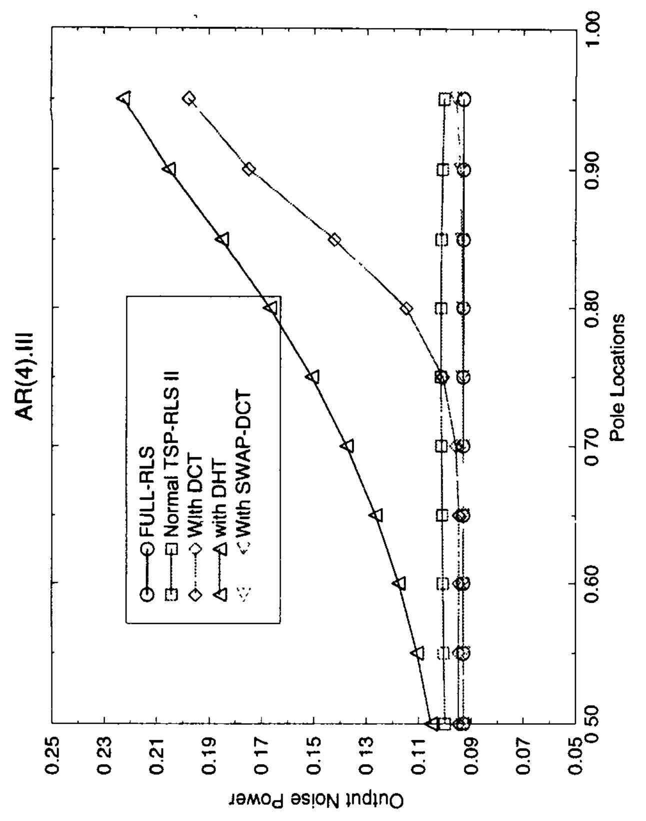
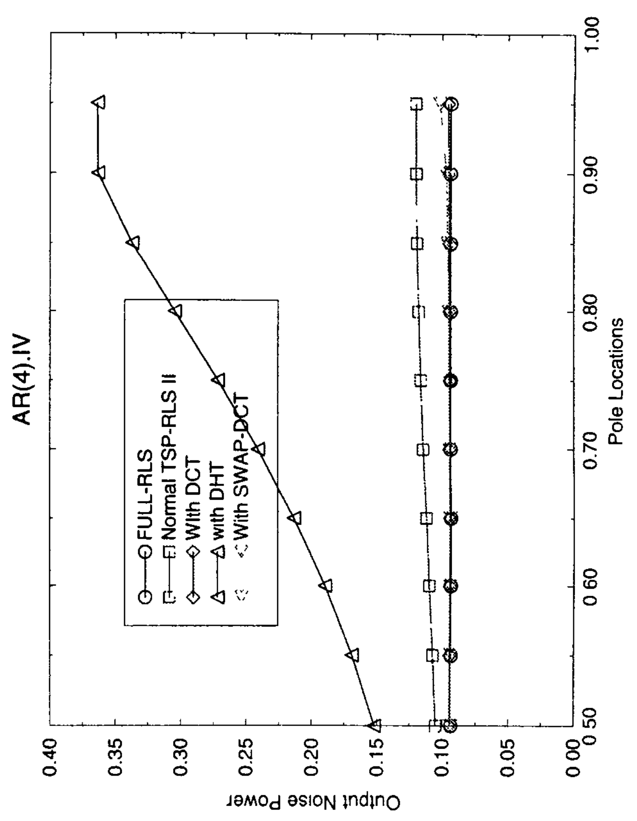
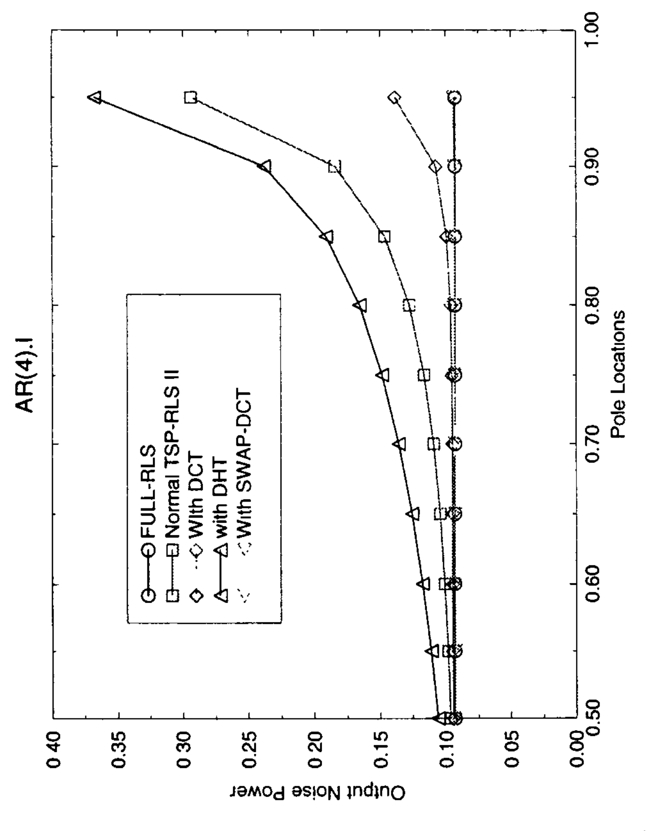


Figure 9: Spectrum of (a) the Normal DCT domain and (b) the SWAP-DCT domain.







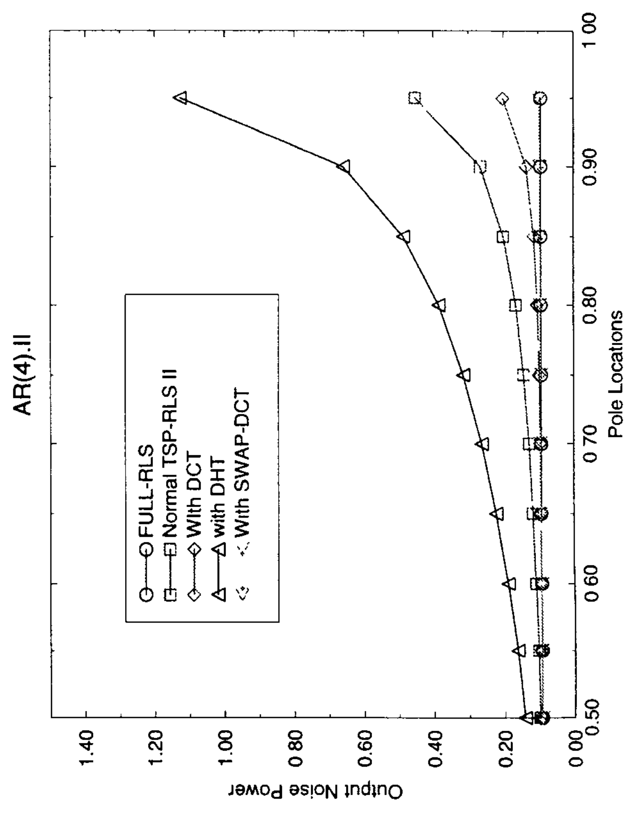


Figure 10: Simulation result of AR(4).I, II, III, IV with preprocessing schemes.

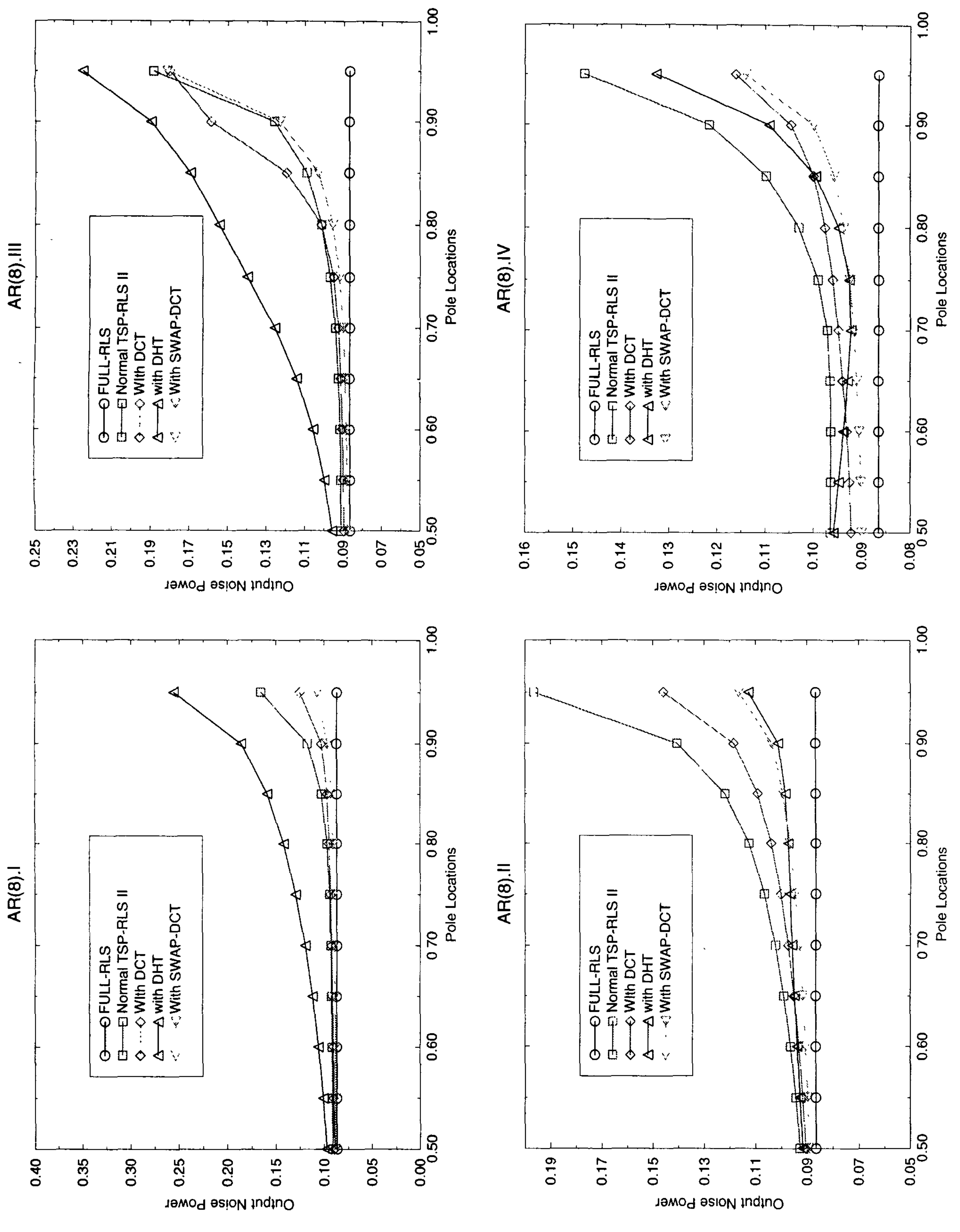


Figure 11: Simulation result of AR(8).I, II, III, IV with preprocessing schemes.

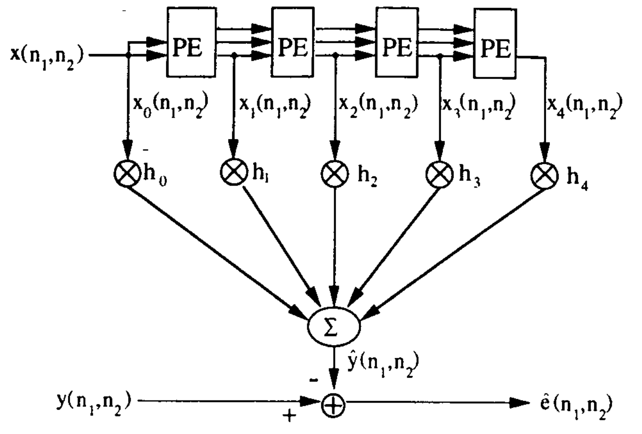


Figure 12: Block diagram of the McClellan Transformation.

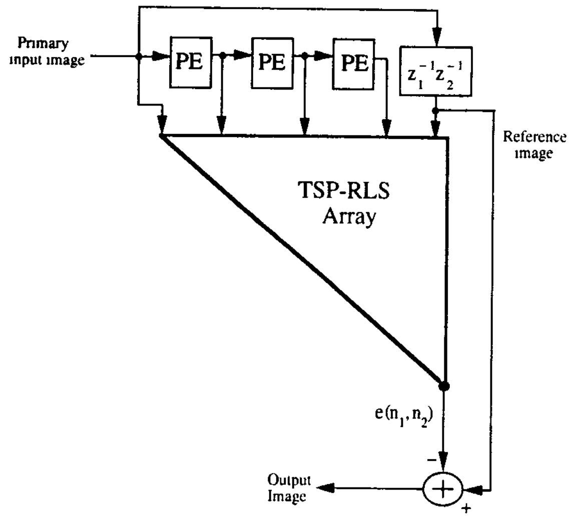


Figure 13: Block diagram of the TDALE.

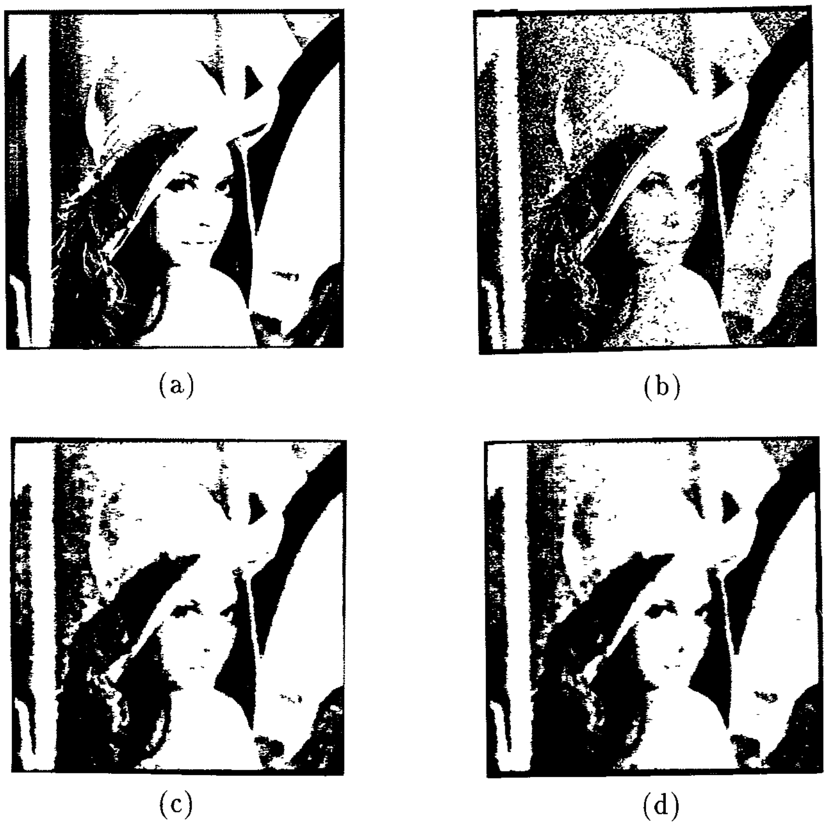


Figure 14: (a) Original LENA image. (b) Noisy input image with SNR=3.7 dB (noise variance = 1000). (c) Output of TDALE with full-size QRD-RLS array (SNR=9.2 dB). (d) Output of TDALE with TSP-RLS array (SNR=10.0 dB).